METHODS FOR DESIGN AND EVALUATION OF PARALLEL COMPUTING SYSTEMS (The PISCES Project)

Terrence W. Pratt
Robert Wise (MS candidate)
Mary Jo Haught (MS candidate)

Virginia Institute for Parallel Computation
Department of Computer Science
University of Virginia
and
ICASE
NASA Langley Research Center

Summary

The PISCES project started in 1984 at the University of Virginia and ICASE under the sponsorship of the NASA CSM program. A PISCES 1 programming environment and parallel Fortran were implemented in 1984 for the DEC VAX (using UNIX processes to simulate parallel processes). This system was used at ICASE and the University of Virginia for experimentation with parallel programs for scientific applications and AI (dynamic scene analysis) applications. PISCES 1 was ported to a network of Apollo workstations by N. Fitzgerald.

Long Term Objective

- Develop new methods for the EVALUATION of parallel computer architectures:
  - — For large-scale scientific software systems used by NASA
  - — Leading to the DESIGN of better parallel systems
- Problem areas:
  - — Variety of available parallel architectures
  - — Software layers impact performance
  - — Performance on SYSTEMS is more important than performance on individual programs
  - — Existing Fortran software base

Long Term Objective

The long term objective of the PISCES project is to develop better ways to evaluate the new parallel computer systems that are coming on the market. The particular target is evaluation of the effectiveness of these parallel systems for the large- scale scientific software systems of interest to NASA. The ultimate goal of the work is to influence the design of the next generation of parallel systems to better meet the needs of NASA software systems.

The major problem areas in evaluating the emerging commercial parallel machines are several. First, the large variety of different parallel architectures available makes evaluation difficult. To reprogram several large NASA systems for each different architecture in order to evaluate performance differences would be expensive and time-consuming.

Second, evaluation of these machines should be in the context of the same sort of programming environments, using Fortran or similar high-level languages, that have traditionally been available on sequential machines. These software layers have an impact on performance that is important to evaluate.

Third, the performance of parallel machines on large NASA software systems is the most important basis for evaluation. Performance on small test programs or on particular algorithms is not of as much interest.

Fourth, the evaluation must take into account the amount of reprogramming of the existing NASA software base that would be required to effectively use these parallel machines. Most of this existing code is in Fortran.

Approach: Experimental

- Build a parallel environment for scientific applications
- Carefully define the software-provided "virtual machine"
- Implement on a variety of machines
- Experiment with language designs (for new code)
- Experiment with porting large systems (for old codes)
- Measure performance differences

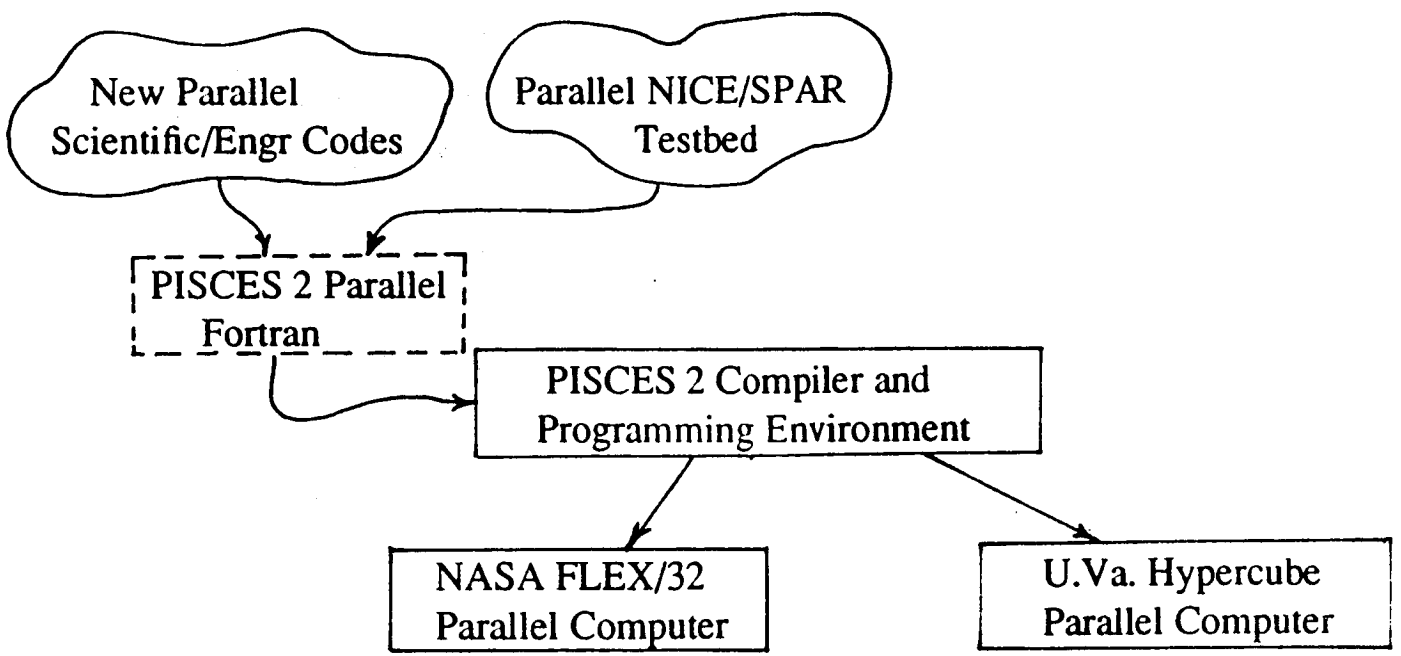
Approach: Experimental

The PISCES project goal is to build a testbed programming environment to support the evaluation of a large range of parallel architectures. This environment, named PISCES, is intended to be Fortran based.

The conceptual basis of the PISCES environment lies in a focus on careful definition of the underlying "virtual machine" provided by the PISCES system for the user. The goal is to implement the same virtual machine on each target architecture, by porting the PISCES software to each machine.

To experiment with Fortran extensions for expressing parallelism, the PISCES environment is designed so that new Fortran extensions can be easily implemented. To experiment with the porting of existing NASA Fortran-based codes, the PISCES environment is designed to allow Fortran codes to be run on various parallel architectures with minimal change. The PISCES environment provides support for automatic collection of performance data, so that performance of large systems may be easily monitored and measured on different architectures.

New Machine Independent Parallel Fortran Implemented on NASA FLEX/32



Result: New parallel algorithms and parallel versions of existing sequential codes may be easily evaluated on the FLEX/32

Summary of Progress

Major results of the PISCES project include an implementation for the NASA Flexible FLEX/32 parallel computer of the PISCES 2 parallel programming environment. PISCES 2 includes a set of extensions to Fortran for parallel computation.

The PISCES 2 system is fully operational on the CSM FLEX/32. The CSM NICE/SPAR testbed system is being modified for parallel operation. An implementation of the testbed in Pisces Fortran on the FLEX/32 is in progress.

The PISCES 2 system is being ported to an NCube/ten hypercube parallel computer at the University of Virginia. The NCube machine has 113 processors with sophisticated parallel graphics and parallel disk subsystems.

The PISCES 2 system on the FLEX/32 provides a useful software base for writing and evaluating new parallel Fortran codes and for modifying and evaluating existing codes for a parallel environment.

Results

- Design of PISCES 2 parallel programming environment: COMPLETED
- Implemention on NASA FLEX/32: COMPLETED
- IN PROGRESS:
  - — NICESPAR testbed port
  - — New parallel algorithms
  - — Hypercube implementation

Results

As noted on the previous slide, the design and implementation of the PISCES 2 system is complete and the system is installed on the CSM FLEX/32.

The port of the NICE/SPAR testbed to PISCES 2 is in progress. The port of PISCES 2 to the U.Va. NCube/ten hypercube is also in progress. Researchers at ICASE, Duke, and U.Va. are using the FLEX/32 system to implement and evaluate new parallel algorithms.

PISCES 2: Main Concepts

- Architecture Independent Virtual Machine
- Multiple Grain Sizes of Parallel Operation
- Clustered Architecture
- Operating System = Static Set of Tasks in Each Cluster
- Dynamic User Task Initiation; Asynchronous Message Passing
- 'Forces' (a la Harry Jordan) with Shared Variables
- Programmer Control of Virtual Machine to Hardware Mapping

PISCES 2: Main Concepts

The PISCES 2 system provides a rich environment for experimenting with parallel programming concepts. The main concepts are:

1. The virtual machine is relatively independent of the underlying architecture, so that programs in Pisces Fortran are not written using constructs peculiar to one parallel architecture.

2. The virtual machine provides several granularities of parallel operation. In PISCES 2 parallel operation is provided between large-grain "clusters of tasks" and "tasks within a cluster" and medium-grain loop iterations and arbitrary program segments.

3. The virtual machine architecture is "clustered", with each cluster representing a group of processing and memory resources of the underlying machine. The cluster organization is static within a single run, but it may be varied between runs.

4. The operating system is represented as a static set of tasks that run within each cluster of the virtual machine. User tasks invoke operating system functions by sending messages to these operating system "controller" tasks.

5. User programs are represented at the top-level by a set of tasks that are dynamically initiated and terminated during a run. Tasks communicate using asynchronous message passing. Message passing queues are infinite (to available memory), and the receiving task may "accept" a message in a variety of different ways, or never.

6. Many of the "FORCE" constructs developed by Harry Jordan are included in PISCES Fortran. These constructs provide medium-grain parallelism, including parallel execution of loop iterations, subroutine calls, and arbitrary program segments. Synchronization mechanisms include barriers and critical regions with lock variables. Communication is via shared variables in COMMON blocks.

7. The programmer controls the mapping of hardware resources to the "clusters" of the PISCES 2 virtual machine. Each run of a program may be configured differently. Currently the mapping consists of assignment of a group of processors to each cluster for running tasks and "forces".

PISCES 2 Implementation

- Extensions to Fortran 77 for parallelism
- Preprocessor: Pisces Fortran ---> standard F77
  - — In-line expansion of parallel constructs
  - — Run-time library
- Configuration Environment
  - — Configure virtual machine (VM)
  - — Map hardware PE's to VM clusters
  - — Create loadfiles to download
  - — Choose run-time options: tracing, timelimit, etc.

PISCES 2 Implementation

The PISCES 2 system as implemented on the CSM FLEX/32 consists of several software components:

1. *Preprocessor.* The extensions to Fortran 77 that form the Pisces Fortran language are implemented with a preprocessor that converts Pisces Fortran to standard Fortran 77, which is then compiled with the FLEX Fortran compiler. The preprocessor converts many parallel constructs to in-line Fortran code. The core parallel constructs are implemented with a small run-time library of routines called from the Fortran code.

2. *Configuration environment.* Before each individual run of a parallel program, the user works in the PISCES "configuration environment" to choose the configuration options for that particular run. These options include the choice of the number of clusters and the cluster numbering, choice of the mapping of hardware resources to these clusters, creation of loadfiles for downloading to hardware PE's, and choice of run-time options such as time limits, tracing options, etc. These options are chosen with a series of menus and prompts. The user does not need to know FLEX OS commands for this activity.

Configurations may be saved in files for later editing/reuse in other runs of the same or different programs. Thus a user can build a library of configurations for use in comparative performance evaluation studies.

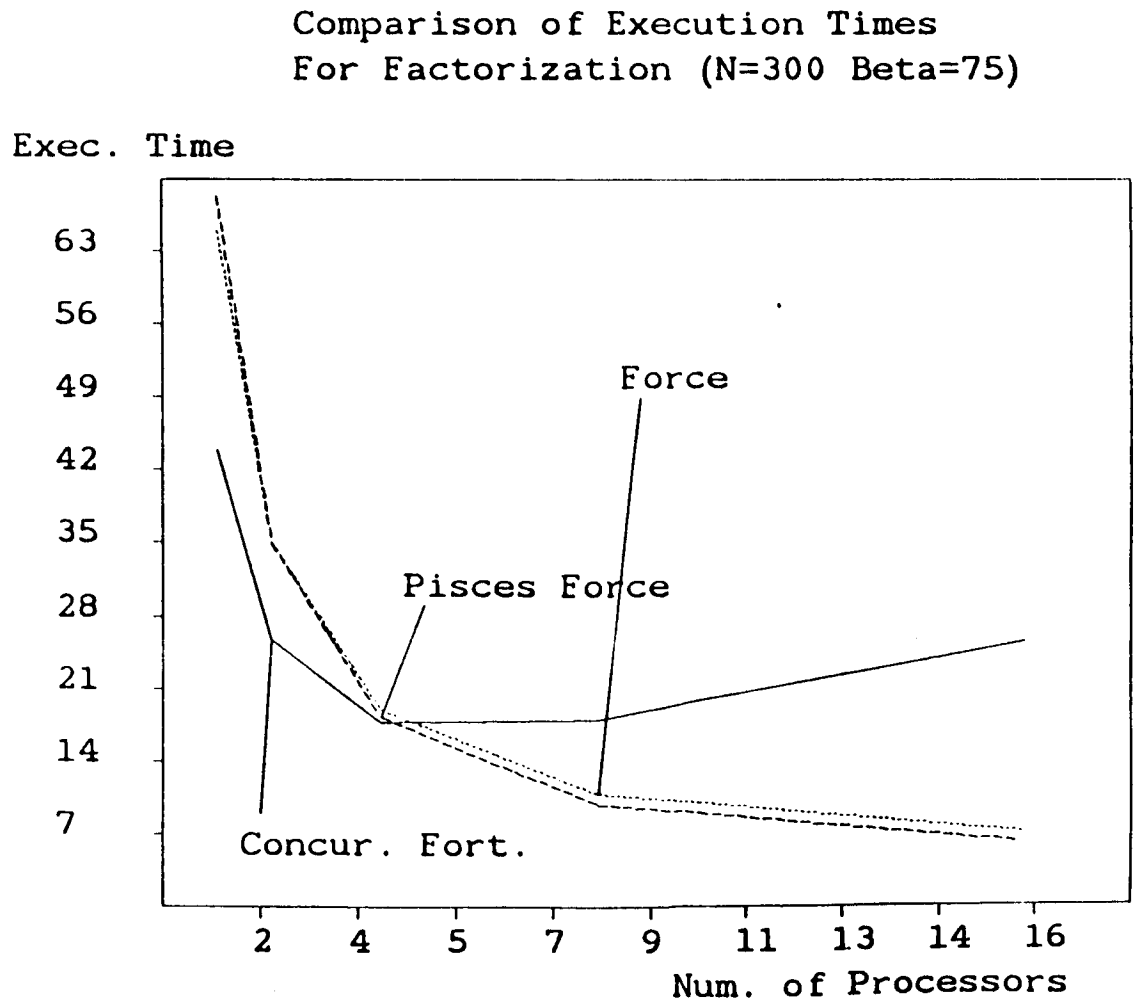
PISCES 2 Implementation (cont)

- Execution Environment:
  - — Control execution: initiate/kill tasks, send messages
  - — Monitor execution: observe system state in real-time
  - — Trace options: save events/times in file or display
- Postprocessing: massage trace files

PISCES 2 Implementation (continued)

3. *Execution environment.* When the user has finished choosing the configuration, execution of a parallel run may be initiated from the configuration environment. When the program begins execution on the FLEX parallel processors, a PISCES "execution environment" provides a menu of commands that allow the user to interact in real-time with the running program. Options include the ability to initiate or kill running tasks, send messages to tasks, monitor execution in real-time, save trace data about key events in trace files, and view the system state (including detailed information about storage management and message queue contents.)

4. *Postprocessing of trace files.* Trace information saved during a run may be processed off-line after the run to obtain detailed information for timing and debugging parallel program components.


Comparison of Execution Times
For Factorization (N=300 Beta=75)
Exec. Time
63
56
49
42
35
28
21
14
7
Force
Pisces Force
Concur. Fort.
2
4
5
7
9
11
13
14
16
Num. of Processors

Performance Comparisons

Conventional wisdom would suggest that machine dependent software provided by the manufacturer of a parallel system would provide better performance than systems such as PISCES 2 or Jordan's FORCE that are implemented as software layers on top of the vendor's software. These performance curves from a study by Prof. Merrell Patrick and Mark Jones of Duke University illustrate that this conventional wisdom can be wrong.

This graph from a Choleski factorization algorithm coded in Flexible's Concurrent Fortran, Jordan's FORCE, and Pisces Fortran shows that the software overhead in Concurrent Fortran quickly dominates this computation, while the software overhead in both PISCES 2 and Jordan's FORCE remains relatively low as the number of processors increases.

These performance measurements are preliminary. More extensive performance measurements and performance tuning of PISCES 2 are in progress.

Publications and Presentations

[1] Pratt, T. "PISCES: An Environment for Parallel Scientific Computation," *IEEE Software*, 2, 4, July 1985. (Also ICASE Report 85-12, February 1985.)

[2] Fitzgerald, N. *Implementation of a Parallel Programming Environment*, M.S. Thesis, Computer Science Dept., U. of Virginia, August 1985.

[3] Patrick, M. and Pratt, T. "Communications Oriented Programming of Parallel Iterative Solutions of Sparse Linear Systems," *Communications in Applied Numerical Methods*, vol. 2, 255-261, 1986.

[4] Pratt, T. "Finding the Right Virtual Machine for Parallel Applications Programming," presentation at *Workshop on Performance Efficient Parallel Programming* (September 1986, Seven Springs, PA, sponsored by NSF and Carnegie-Mellon University).

[5] Pratt, T. "The PISCES 2 Parallel Programming Environment," presentation at *Parallel Languages and Environments Workshop* (November 1986, Williamsburg, VA, sponsored by ICASE and NASA).

[6] Pratt, T. "Short Course: Introduction to the PISCES 2 Parallel Programming Environment," presented at NASA LaRC/ICASE (March 1987) and U. of Virginia (April 1987).

[7] Pratt, T. *PISCES 2 User's Manual (Version 2)*, ICASE Interim Report 2, July 1987.

[8] Pratt, T. "The PISCES 2 Parallel Programming Environment," *Proceedings 1987 International Conference on Parallel Processing*, St. Charles, IL, August 1987, 439-445 (also ICASE Report 87-38, July 1987).